Title	組合せの効率的な生成法 (計算機科学とアルゴリズムの数理的基礎とその応用)
Author(s)	Shimizu, Toshihiro; Fukunaga, Takuro; Nagamochi, Hiroshi
Citation	数理解析研究所講究録 (2011), 1744: 99-106
Issue Date	2011-06
URL	http://hdl.handle.net/2433/170966
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第1744巻 2011年 99-106

組合せの効率的な生成法

清水俊宏　　福永拓郎　　永持仁

京都大学大学院情報学研究科数理工学専攻
{shimizu,takuro,nag}@amp.i.kyoto-u.ac.jp

Abstract: An unranking algorithm of a finite set S is an algorithm such that, given a number in $\{0,1,\ldots,|S|-1\}$, it returns an element of S which is associated with the number. In this paper, it is allowed to associate a number to any element in S whenever the distinct elements are associated with different numbers. A ranking algorithm is the reverse of an unranking algorithm. In this paper, we present two unranking algorithms for the set of all m-element subsets of an n-element set. One runs in $O(n\log m)$ time, and the other runs in $O(m^{3m+3})$ time. We also show that they both have ranking algorithms with the same running times.

1 Introduction

For a positive integer n, let $\langle n\rangle$ denote $\{0,1,\ldots,n-1\}$. A *ranking algorithm* of a finite set S is defined as an algorithm such that, given an element of S, it returns an integer in $\langle|S|\rangle$ which is associated with the element. It is allowed to associate an integer in $\langle|S|\rangle$ to any element in S whenever the distinct elements are associated with different integers. In other word, we can order the elements in S arbitrarily to obtain an efficient ranking algorithm. If the algorithm returns $r\in\langle|S|\rangle$ as the answer to an element in S, r is called the *rank* of the element and the element is called *the r-th element* (with respects to the algorithm). An *unranking algorithm* is the reverse of the ranking algorithm; Given an integer in $\langle|S|\rangle$, it returns an element of S.

This paper discusses ranking and unranking algorithms of a fundamental finite set. For non-negative integers n and m with $n\geq m$, let

$$C(n,m)$$
$$=\{(a_1,a_2,\ldots,a_m):0\leq a_1<a_2<\cdots<a_m\leq n-1\}.$$

$C(n,m)$ is equivalent to the set of all m-element subsets of the n-element set $\langle n\rangle$, and hence $|C(n,m)|=\binom{n}{m}$.

In this paper, we evaluate the running time of algorithms by the numbers of arithmetic operations. Since the largest rank in $C(n,m)$ is $\binom{n}{m}-1=\Theta(n^m)$, algorithms discussed in this paper cannot avoid dealing with numbers represented by $O(m\log n)$ bits. We assume that each operation of such numbers can be done in $O(1)$ time.

A naive idea for ranking and unranking of $C(n,m)$ is to order the elements in the reverse lexicographic order. In this order, the rank of $(a_1,a_2,\ldots,a_m)\in C(n,m)$ is $\sum_{i=1}^{m}\binom{n-a_i-1}{m-i+1}$. Since computation of $\binom{n-a_i-1}{m-i+1}$ needs $O(m-i+1)$ time, there is an $O(m^2)$-time ranking algorithm with this order. On the contrary, the rank r of $(a_1,a_2,\ldots,a_m)\in C(n,m)$ in the reverse lexicographic order is at least $\binom{n-1}{m}$ if and only if $a_1=0$. Thus, given the rank r, we can judge whether $a_1=0$ or not in $O(m)$ time. When $a_1=0$, $a_2=1$ if and only if $r\geq\binom{n-1}{m}+\binom{n-2}{m-1}$. This condition can be checked in $O(1)$ time by computing $\binom{n-2}{m-1}$ from $\binom{n-1}{m}$. Repeating such operations gives an $O(n)$-time unranking algorithm with the reverse lexicographic order.

The aim of this paper is to propose better unranking algorithms of $C(n,m)$. In fact, we propose two new unranking algorithms of $C(n,m)$. At first, we propose an easy recursive algorithm running in $O(m\log n)$ time. It is based on the observation that $C(n,m)$ is equivalent to $\cup_{i=0}^{m}C(\lfloor n/2\rfloor,i)\times C(\lceil n/2\rceil,m-i)$. The algorithm defines an order different from the lexicographic order and its reverse; $(a_1,a_2,\ldots,a_m)$ is former than $(a'_1,a'_2,\ldots,a'_m)$ if $|\{a_1,a_2,\ldots,a_m\}\cap\{0,1,\ldots,\lfloor n/2\rfloor\}|<|\{a'_1,a'_2,\ldots,a'_m\}\cap\{0,1,\ldots,\lfloor n/2\rfloor\}|$. The deteail is explained in Section 2. In the next, we

propose an $O(m^{3m+3})$-time unranking algorithm. Since its running time depends on m only, this is a huge improvement when m is small. The order defined by this algorithm implicitely is also different from the lexicographic order. This algorithm is designed based on a new insight on the structure of $C(n,m)$. We also show that each of our unranking algorithms admits a ranking algorihtm with the same running time.

Let us mention the background of this work. It is a fundamental topic in computer science to study algorithms to generate all objects in certain sets one by one, which are sometimes called enumeration algorithms. In fact, a number of enumeration algorithms are known for $C(n,m)$ (for example, [6, 11, 21]). They have many applications such as data mining [1, 2], artificial intelligence [7, 22], operations research [3, 10, 14, 16, 17, 24], and bioinformatics [4, 9]. However there are several situations in which they are not useful. For example, let us consider investigating certain objects one by one. During the investigation, we sometimes need to retrieve an object which has been already investigated. To do this, an enumeration algorithm needs to enumerate the objects from the beginning again whereas an unranking algorithm can directly generate the object if its rank, which can be obtained by a ranking algorithm, is remembered. Considering this fact, ranking and unranking algorithms can be regarded as more advanced algorithms than enumeration algorithms. In fact, if we have an unranking algorithm running in $O(t)$ time, then we can enumerate objects in $O(t)$ time per an object. We expect that efficient ranking/unranking algorithms of fundamental objects are applied in many fields by this reason. One example of such applications is the work due to Imada et al. [9]. They designed an algorithm for enumerating stereoisomers of chemical compounds, in which an unranking algorithm of $C(n,m)$ with $m \leq 4$ is used as a subroutine.

Ranking and unranking algorithms of a set S can be regarded as an algorithmic implementation of a bijection between $\langle |S| \rangle$ and S. If efficient ranking and unranking algorithms of S are available, we can use an integer in $\langle |S| \rangle$ as a compact representation of an element in S. It has been pointed in [20] that this feature of ranking and unranking algorithms can be applied for random sampling of $|S|$; Generate an integer in $\langle |S| \rangle$ uniformly at random, and unrank it by an unranking algorithm. Then elements of S are sampled uniformly.

Let us mention the previous researches on ranking and unranking algorithms. As for $C(n,m)$, Liebehenschel [15] presented an average-case analysis of ranking and unranking algorithms for the set of lexicographically ordered words, which extends $C(n,m)$. Several parallel unranking algorithms of $C(n,m)$ are discussed in [12, 13]. To the best of our knowledge, there are no known algorithms which outperform our algorithms. Ranking and unranking algorithms are studied for various objects such as permutations [18, 19], trees [5, 20] and B-trees [8]. The most related works among them are about permutations of m elements chosen from an n-element set, which are discussed in Mareš and Straka [18] and Myrvold and Ruskey [19]. In particular, the approach of Myrvold and Ruskey [19] is similar with ours in the fact that they did not persist on the lexicograhic order. Although both of them proposed $O(m)$ ranking and unranking algorithms, their algorithms cannot be applied to $C(n,m)$.

The rest of this paper is organized as follows. We present an $O(m \log n)$-time ranking and unranking algorithms of $C(n,m)$ in Section 2, and $O(m^{3m+3})$-time ranking and unranking algorithm of $C(n,m)$ in Section 3. In Section 4, we conclude the paper.

2 $O(m \log n)$-time ranking and unranking algorithms

We first present an $O(m \log n)$-time unranking algorithms of $C(n,m)$.

For given n, let $\tilde{n} = \lfloor n/2 \rfloor$. Moreover we define $T_i = \{e \in C(n,m) : |e \cap \langle \tilde{n} \rangle| = i\}$ for any integer i satisfying $0 \leq i \leq m$, where we here let $|e \cap \langle \tilde{n} \rangle|$ indicate the number of components of e contained by $\langle \tilde{n} \rangle$. Obviously $T_0, T_1, \ldots, T_m$ are pairwise disjoint and $C(n,m) = \cup_{i=0}^{m} T_i$.

Let $(a_1, a_2, \ldots, a_m) \in T_i$. Then $(a_1, a_2, \ldots, a_i) \in C(\tilde{n}, i)$ and $(a_{i+1} - \tilde{n}, a_{i+2} - \tilde{n}, \ldots, a_m - \tilde{n}) \in C(n - \tilde{n}, m - i)$. That is to say, each element in T_i is the concatenate of a vector in $C(\tilde{n}, i)$ and a vector constructed from one in $C(n - \tilde{n}, m - i)$ by adding $\tilde{n}$ to all components. This fact implies that an unranking algorithm of $C(n,m)$ can be constructed from those of $C(\tilde{n}, i)$, $i = 0, 1, \ldots, \tilde{n}$ and

$C(n-\tilde{n}, i)$, $i = 0, 1, \ldots, n-\tilde{n}$, which is described precisely below.

Let $\theta_0 = 0$ and $\theta_i = \sum_{j=0}^{i-1} |T_j|$ for $1 \leq i \leq m+1$, where $|T_j| = \binom{\tilde{n}}{j} \cdot \binom{n-\tilde{n}}{m-j}$. Our unranking algorithm associates the rank r with a vector in T_i if $\theta_i \leq r < \theta_{i+1}$. In other words, we order vectors in $C(n, m)$ from those in T_0 to in T_m.

The ordering in each T_i is defined from the unranking algorithms of $C(\tilde{n}, i)$ and $C(n-\tilde{n}, m-i)$ by the following rule. For $0 \leq r' \leq |T_i|$, define r_1 and r_2 as the remainder and quotient when r' is divided by $|C(\tilde{n}, i)|$, respectively (i.e., $r' = r_2|C(\tilde{n}, i)| + r_1$ and $0 \leq r_1 < |C(\tilde{n}, i)|$). Recall that $|C(\tilde{n}, i)| = \binom{\tilde{n}}{i}$. Let $(a_1, a_2, \ldots, a_i) \in C(\tilde{n}, i)$ be the r_1-th vector of $C(\tilde{n}, i)$ and $(b_1, b_2, \ldots, b_{m-i})$ be the r_2-th vector of $C(n-\tilde{n}, m-i)$, respectively. We define the r'-th vector of T_i from them as $(a_1, a_2, \ldots, a_i, b_1 + \tilde{n}, b_2 + \tilde{n}, \ldots, b_{m-i} + \tilde{n})$.

Entire our algorithm is described as follows.

Algorithm UNRANKING(n, m, r)

Input: Non-negative integers n and m with $m \leq n$, and a rank $r \in \langle \binom{n}{m} \rangle$

Output: A vector in $C(n, m)$

Step 1: If $m = 0$, then return $\emptyset$. If $n = m$, then return $(0, 1, \ldots, n-1)$.

Step 2: Compute the index i such that $\theta_i \leq r < \theta_{i+1}$, where $\theta_0 = 0$ and $\theta_i = \sum_{j=0}^{i-1} \binom{\tilde{n}}{j} \cdot \binom{n-\tilde{n}}{m-j}$, $1 \leq i \leq m$.

Step 3: Compute the remainder r_1 and the quotient r_2 when $(r - \theta_i)$ is divided by $\binom{\tilde{n}}{i}$. Compute the r_1-th vector $(a_1, a_2, \ldots, a_i)$ of $C(\tilde{n}, i)$ and the r_2-th vector $(b_1, b_2, \ldots, b_{m-i})$ of $C(n-\tilde{n}, m-i)$ by calling UNRANKING($\tilde{n}, i, r_1$) and UNRANKING($\tilde{n}, m-i, r_2$).

Step 4: Return $(a_1, a_2, \ldots, a_i, b_1 + \tilde{n}, \ldots, b_{m-i} + \tilde{n})$.

Now let us analyze the time $t(n, m)$ to compute the r-th vector of $C(n, m)$ by UNRANKING(n, m, r). Obviously $t(n, 0) = O(1)$ and $t(n, n) = O(1)$ by Step 1.

θ_{i+1} can be computed from θ_i in $O(1)$ time, and hence all of $\theta_0, \theta_1, \ldots, \theta_m$ can be computed in $O(m)$ time in total. Thus all operations except calling UNRANKING($\tilde{n}, i, r_1$) and UNRANKING($\tilde{n}, m-i, r_2$) take $O(m)$ time. It implies that

$$t(n, m) = t(\lfloor n/2 \rfloor, i) + t(\lceil n/2 \rceil, m-i) + O(m).$$

From this, $t(n, m) = O(m \log n)$ can be proven by the induction on n and m.

UNRANKING(n, m, r) can be modified into a ranking algorithm as follows. Let $(a_1, a_2, \ldots, a_m) \in C(n, m)$ be an input to the algorithm. In Step 2, set i to $|\{a_1, a_2, \ldots, a_m\} \cap \langle \tilde{n} \rangle\}|$. Let $a' = (a_1, a_2, \ldots, a_i)$ and $a'' = (a_{i+1} - \tilde{n}, a_{i+2} - \tilde{n}, \ldots, a_m - \tilde{n})$. In Step 3, call the ranking algorithms with inputs $(\tilde{n}, i, a')$ and $(\tilde{n}, m-i, a'')$ recursively. Let r_1 and r_2 be the obtained ranks of a' and a''. Then the rank of $(a_1, a_2, \ldots, a_m)$ is $\theta_i + r_1|C(\tilde{n}, i)| + r_2$. The running time of this ranking algorithm can be derived similarly.

As a consequence, we have the next theorem.

Theorem 1. *There exist ranking and unranking algorithms of $C(n, m)$ which run in $O(m \log n)$ time.*

3 $O(m^{3m+3})$-time ranking and unranking algorithms

3.1 Sketch of idea

In this section, we show that there exist ranking and unranking algorithms of $C(n, m)$ running in $O(m^{3m+3})$ time. Our proof is based on the induction on m; We prove that, from $O(t_{m'})$-time ranking/unranking algorithms for $C(n', m')$, $m' < m$, it is possible to construct an $O(t_{m'} + m^{3m+2})$-time ranking/unranking algorithm for $C(n, m)$. In this subsection, we sketch an idea behind our proof.

It is known as a fundamental theorem that $\binom{n}{m} = \binom{n}{m-1} + \binom{n-1}{m-1}$. Its proof uses the fact that there exists a bijection between $C(n, m)$ and $C(n, m-1) \cup C(n-1, m-1)$. From this relationship, we observe that it suffices to consider only n and m that are coprime (Lemma 7 and Theorem 2).

Now put balls indexed by the numbers in $\langle n \rangle$ in a circle, and represent a vector $(a_1, a_2, \ldots, a_m) \in C(n, m)$ by filling the balls indexed by $a_1, a_2, \ldots, a_m$ as in Figure 1. We then divide $C(n, m)$ into groups such that $a \in C(n, m)$ and $b \in C(n, m)$ are in the same group if the representation of a becomes the same with that of b after removing the indices of balls (see the rightmost figure of Figure 1). The set of vectors each of which is lexicographically smallest in its group is denoted by $R(n, m)$ in the next subsection. If n and m are coprime, each group has exactly n vectors.

Hence there exists a bijection between $C(n,m)$ and $R(n,m) \times \langle n \rangle$, meaning that a ranking/unranking algorithm can be constructed from one for $R(n,m)$ (Lemma 6).

In the ball representation of $a \in R(n,m)$ without indices, define the distance between two filled balls as i if there are $i-1$ balls between them. Instead of balls, locate distances between two adjacent filled balls in a circle. $SL(n,m)$ defined in the next subsection is a set of vectors representing them uniquely (see Figure 2). $SL(n,m)$ is essentially equivalent to $R(n,m)$ (Lemma 5).

Let $a \in SL(n,m)$, and $b_1, b_2, \ldots, b_{m'}$ be the distinct values in the components of a, where $m' \leq m$ because a is an m-dimensional vector. We suppose that $b_1 < b_2 < \cdots < b_{m'}$. The number of possible places of $b_1, b_2, \ldots, b_{m'}$ in a is at most $O(m^m)$, and hence we can enumerate all of them. In the proof of Lemma 4, a vector representing the places is called *position vector* p. If a position vector is given, we know how many times b_i appears in a; Assume that b_i appears e_i times for $i = 1, 2, \ldots, m'$. Then $\sum_{i=1}^{m'} b_i e_i = n$. The set $MES(n, m', e)$ of vectors is defined as $\{(b_1, b_2, \ldots, b_{m'}) : \sum_{i=1}^{m'} b_i e_i = n, 1 \leq b_1 < b_2 < \cdots < b_{m'} \leq n\}$ in the next subsection. This indicates a bijection between $SL(n,m)$ and $\cup_{j=1}^{\ell} MES(n, m^j, e^j)$ where m^j and e^j are determined by the enumerated position vectors p^j, $j = 1, 2, \ldots, \ell$. From this fact, we can construct a ranking/unranking algorithm for $SL(n,m)$ from one for $MES(n, m^j, e^j)$ (Lemma 4). We then carefully transform $MES(n, m^j, e^j)$ into $C(n', m')$, $m' < m$, meaning that a ranking/unranking algorithm for $C(n', m')$, $m' < m$, gives one for $MES(n, m^j, e^j)$ (Lemmas 1, 2 and 3).

3.2 Algorithms

Let us present an exact proof of our algorithm. Due to the space limitation, several proofs are omitted. From now on, let us suppose that there exist $O(t_{m'})$-time ranking and unranking algorithms for $C(n', m')$ with any n and $m' < m$.

We define a finite set $S(n,m)$ by $S(n,m) = \{(a_1, a_2, \ldots, a_m) : 1 \leq a_1, a_2, \ldots, a_m \leq n, \sum_{i=1}^{m} a_i = n\}$. Notice that the range of components of vectors in $S(n,m)$ is different from that in $C(n,m)$. We also note that $|S(n,m)| = \binom{n-1}{m-1}$.

We can observe that a ranking/unranking algorithm of $C(n, m-1)$ gives a ranking/unranking algorithm of $S(n,m)$.

Lemma 1. *There exist ranking and unranking algorithms of $S(n,m)$ running in $t_{m-1} + O(m)$ time.*

Proof.
Bijection: Let $(a_1, a_2, \ldots, a_m) \in S(n,m)$. Moreover let $b_i = (\sum_{j=1}^{i} a_j) - 1$ for $i = 1, 2, \ldots, m$, and $b = (b_1, b_2, \ldots, b_{m-1})$. It then follows that

$$0 \leq b_1 < b_2 < \cdots < b_{m-1} < b_m = n - 1,$$

and hence $b \in C(n-1, m-1)$. This defines a bijection from $S(n,m)$ to $C(n-1, m-1)$. In this proof, let $f : S(n,m) \to C(n-1, m-1)$ denote this bijection.

Unranking: For given rank $r \in \langle |S(n,m)| \rangle$, compute the r-th vector $b = (b_1, b_2, \ldots, b_{m-1})$ in $C(n-1, m-1)$ by the unranking algorithm of $C(n-1, m-1)$ and return $f^{-1}(b) \in S(n,m)$. Since $f^{-1}(b)$ can be computed from b in $O(m)$ time, this unranking algorithm runs in $t_{m-1} + O(m)$ time.

Ranking: For given $a \in S(n,m)$, compute the rank of $f(a) \in C(n-1, m-1)$ by the ranking algorithm of $C(n-1, m-1)$. □

By an m-dimensional positive vector $b = (b_1, b_2, \ldots, b_m)$, we extend $S(n,m)$ to $ES(n,m,b) = \{(a_1, a_2, \ldots, a_m) : 1 \leq a_1, a_2, \ldots, a_m \leq n, \sum_{i=1}^{m} b_i a_i = n\}$. Note that $ES(n,m,b) = S(n,m)$ if $b = (1, 1, \ldots, 1)$. In the next, we consider ranking and unranking algorithms of $ES(n,m,b)$. For integers i and j, we let $i \mid j$ mean that i divides j, i.e., $i \mod j = 0$.

Lemma 2. *There exist ranking and unranking algorithms of $ES(n,m,b)$ running in $t_{m-1} + O(m^2 \beta^{2m-1})$ time for any n and $b = (b_1, b_2, \ldots, b_m)$ such that $\max_{i=1}^{m} b_i \leq \beta$. Moreover $|ES(n,m,b)|$ can be computed in $O(m^2 \beta^{2m-1})$ time.*

For $(b_1, b_2, \ldots, b_m)$, let $MES(n,m,b) = \{(a_1, \ldots, a_m) \in ES(n,m,b) : 1 \leq a_1 < \cdots < a_m \leq n\}$. In the next, we give ranking and unranking algorithms of $MES(n,m,b)$.

Lemma 3. *There exist ranking and unranking algorithms of $MES(n,m,b)$ running in $t_{m-1} + O(m^2 \beta^{2m-1})$ time for any $b = (b_1, b_2, \ldots, b_m)$ such that $\sum_{i=1}^{m} b_i \leq \beta$. Moreover $|MES(n,m,b)|$ can be computed in $O(m^2 \beta^{2m-1})$ time.*

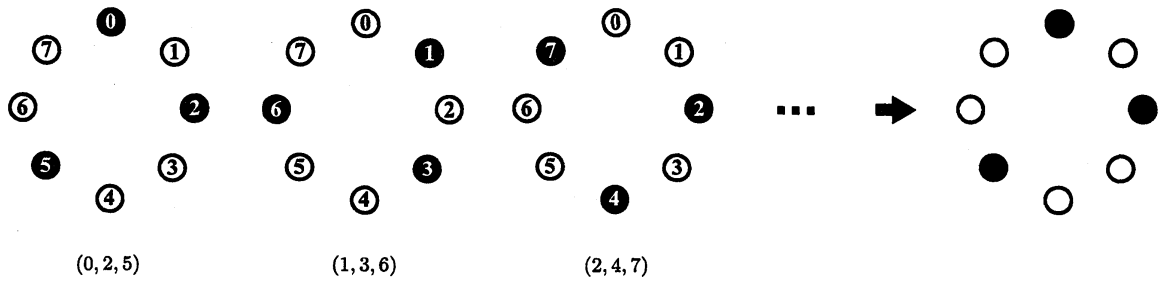

Figure 1: Representation of vectors in $C(n, m)$ by balls in a circle

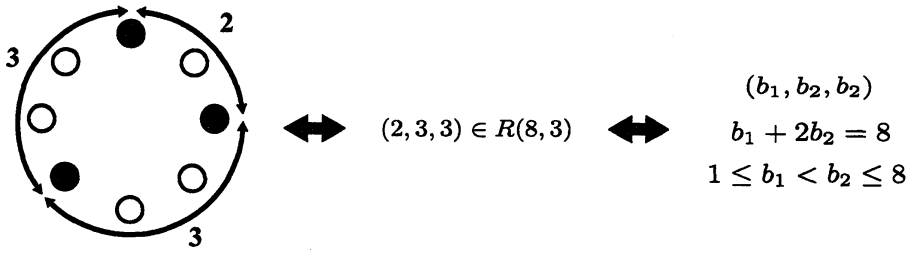

Figure 2: From $R(n, m)$ to $SL(n, m)$, and from $SL(n, m)$ to pairs of position vectors and $MES(n, m', e)$

Proof.
Bijection: From a vector $a = (a_1, a_2, \ldots, a_m) \in MES(n, m, b)$, define $c = (c_1, c_2, \ldots, c_m)$ by $c_1 = a_1$ and $c_i = a_i - a_{i-1}$, $i = 2, 3, \ldots, m$. Let f denote this mapping in this proof. Observe that $c_i \geq 1$ for $i = 1, 2, \ldots, m$. Moreover, define $d = (d_1, d_2, \ldots, d_m)$ by $d_i = \sum_{j=i}^{m} b_j$, $i = 1, 2, \ldots, m$. Since $\sum_{i=1}^{m} a_i b_i = n$ and $a_i = \sum_{j=1}^{i} c_j$ for $i = 1, 2, \ldots, m$, it follows that

$$\sum_{i=1}^{m} c_i d_i = \sum_{i=1}^{m} c_i \sum_{j=i}^{m} b_j = \sum_{j=1}^{m} b_j \sum_{i=1}^{j} c_i = \sum_{j=1}^{m} b_j a_j = n,$$

implying that $c \in ES(n, m, d)$. Hence f is a bijection from $MES(n, m, b)$ to $ES(n, m, d)$.

Unranking: Let $r \in \langle |MES(n, m, b)| \rangle$ be the given rank. From b, we compute d defined as above. Recall that $d_i = \sum_{j=i}^{m} b_j \leq \beta$ for every $i = 1, 2, \ldots, m$. We then compute the r-th vector c of $ES(n, m, d)$ by the unranking algorithm of $ES(n, m, d)$, which runs in $t_{m-1} + O(m^2 \beta^{2m})$ time by Lemma 2. We then compute $f^{-1}(c) \in MES(n, m, b)$ and return it. It is easy to see that the running time of this unranking algorithm is determined by those of the unranking algorithm in Lemma 2.

Ranking: From b, compute d defined as above. From the given $a \in MES(n, m, b)$, Compute $f(a)$, and return the rank of $f(a)$ in $ES(n, m, d)$.

Since $|MES(n, m, b)| = |ES(n, m, d)|$, we can also compute $|MES(n, m, b)|$ in the same running time with the computation of $|ES(n, m, d)|$. □

We say that a vector $(a_1, a_2, \ldots, a_m)$ is a *slide* of another vector $(b_1, b_2, \ldots, b_m)$ whenever there exists $j \in \{0, 1, 2, \ldots, m-1\}$ such that $a_i = b_{(i+j)}$ for all $i \in \{1, 2, \ldots, m\}$ where we let $b_{i+j} = b_{i+j-m}$ if $i + j > m$ for convenience. Let $SL(n, m)$ be the subset of $S(n, m)$ such that $a \in S(n, m)$ belongs to $SL(n, m)$ if and only if a is lexicographically smaller than or equal to any of its slides.

Lemma 4. *There exist ranking and unranking algorithms of $SL(n, m)$ running in $O(t_{m-1} + m^{3m+2})$ time.*

We say that a vector $a = (a_1, a_2, \ldots, a_m)$ is a *rotation* of $b = (b_1, b_2, \ldots, b_m)$ whenever there exists an integer j such that $\{a_1, a_2, \ldots, a_m\} \equiv \{b_1 + j, b_2 + j, \ldots, b_m + j\} \pmod{n}$. Let $R(n, m)$ be the subset of $C(n, m)$ such that $a \in C(n, m)$ belongs to $R(n, m)$ if and only if a is lexicographically smaller than or equal to any of its rotations.

Lemma 5. *If n and m are coprime, then there exist ranking and unranking algorithms of $R(n,m)$ running in $t_{m-1} + O(m^{3m+2})$ time.*

Proof.
Bijection: From $a = (a_1, a_2, \ldots, a_m) \in S(n,m)$, define $b = (b_1, b_2, \ldots, b_m)$ by

$$b_i = \begin{cases} 0 & i = 1, \\ a_1 + a_2 + \cdots + a_{i-1} & 2 \le i \le m. \end{cases}$$

Then $0 \le b_1 < b_2 < \cdots < b_m < n$, i.e., $b \in C(n,m)$, because $a_i \ge 1$ for $i = 1,2,\ldots,m$ and $\sum_{i=1}^{m} a_i = n$. It is easy to see that this transformation from a to b defines a bijection from $S(n,m)$ to $\{(b_1, b_2, \ldots, b_m) \in C(n,m) : b_1 = 0\}$. We denote it by f.

The restriction of f onto $SL(n,m)$ is a bijection from $SL(n,m)$ to $R(n,m)$ when n and m are coprime. For proving this fact, we need to show that the following two conditions hold:

(F1) For each $a \in SL(n,m)$, $f(a) \in R(n,m)$;

(F2) For each $b \in R(n,m)$, there exists $a \in SL(n,m)$ such that $f(a) = b$.

Here we do not prove (F1) and (F2) due to the space limitation.

Unranking: For the given rank $r \in \langle |R(n,m)| \rangle$, compute the r-th vector $a \in SL(n,m)$ by the unranking algorithm of $SL(n,m)$ in Lemma 4, and return $f(a)$. Since $f(a)$ can be computed from a in $O(m)$ time, this algorithm runs in $t_{m-1} + O(m^{3m+2})$ time.

Ranking: The reverse operations of the unranking algorithm give a ranking algorithm of $R(n,m)$. Since $f^{-1}(b)$ can be computed from $b \in R(n,m)$ in $O(m)$ time, this runs in $t_{m-1} + O(m^{3m+2})$ time. □

Lemma 6. *If n and m are coprime, then there exist ranking and unranking algorithms of $C(n,m)$ running in $t_{m-1} + O(m^{3m+2})$ time.*

Proof.
Bijection: Let $a = (a_1, a_2, \ldots, a_m) \in C(n,m)$. We define U_a as the set of rotations of a in $C(n,m)$. First we prove that $|U_a| = n$. We represent $\{a_1, a_2, \ldots, a_m\}$ by A, and $\{(a_1 + j) \bmod n, (a_2 + j) \bmod n, \ldots, (a_m + j) \bmod n\}$ by $A + j$ for an integer j. Notice that each vector in U_a can be defined from each of $A + j$, $j \in \langle n \rangle$. If $|U_a| = n$ is proven, we can see that each of $A + j$, $j \in \langle n \rangle$ defines a distinct vector of U_a. Recall that $R(n,m)$ consists of the vectors $a \in C(n,m)$ such that a is lexicographically smallest in U_a. Hence these imply a bijection from $R(n,m) \times \langle n \rangle$ to $C(n,m)$. We omit the proof of $|U_a| = n$ due to the space limitation.

Unranking: For the given rank $r \in \langle |C(n,m)| \rangle$, let p be the remainder and q be the quotient when r is divided by n. By the unranking algorithm of $R(n,m)$ given in Lemma 5, compute the q-th vector $(a_1, a_2, \ldots, a_m)$ in $R(n,m)$. There exists only one slide of $(a_1 + p, a_2 + p, \ldots, a_m + p) \bmod n$ which is in $C(n,m)$. Hence return it as the r-th vector of $C(n,m)$. This computation needs $t_{m-1} + O(m^{3m+2})$ time.

Ranking: For the given $a = (a_1, a_2, \ldots, a_m) \in C(n,m)$, compute the lexicographically smallest vector $a' = (a'_1, a'_2, \ldots, a'_m)$ in U_a. Then compute the rank r' of a' in $R(n,m)$ by the ranking algorithm in Lemma 5. Moreover, compute $j \in \langle n \rangle$ such that $\{a_1, a_2, \ldots, a_m\} = \{a'_1 + j, a'_2 + j, \ldots, a'_m + j\}$. Return $r'n + j$ as the rank of a. Observe that the running time is dominated by the time for calling the ranking algorithm of $R(n,m)$. □

The next lemma is important to extend the unranking algorithm described in Lemma 6 to any pair of n and m.

Lemma 7. *Let n and m be positive integers. If there exists an unranking (resp., a ranking) algorithm of $C(n-1,m)$ running in $O(t)$ time, and an unranking (resp., a ranking) algorithms of $C(n-1,m-1)$ running in $O(t')$ time, then there exists an unranking (resp., a ranking) algorithms of $C(n,m)$ running in $O(m + \max\{t,t'\})$ time.*

Finally we obtain our main theorem.

Theorem 2. *There exist ranking and unranking algorithm of $C(n,m)$ running in $O(m^{3m+3})$ time.*

Proof. There exists a non-negative integer $m' < m$ such that $n - m' \equiv 1 \pmod m$. Then $n' = n - m'$ and m are coprime. By applying Lemma 7 repeatedly, we can observe that an unranking (resp., a ranking) algorithm of $C(n,m)$ is constructed from unranking (resp., ranking) algorithms of $C(n', m-1), C(n'+1, m-1), \ldots, C(n, m-1)$ and $C(n', m)$. For each of $C(n', m-1), C(n'+1, m-$

$1), \ldots, C(n, m-1)$, we have ranking and unranking algorithms running in t_{m-1} time by the assumption. Lemma 6 tells that there exist ranking and unranking algorithms of $C(n', m)$ running in $t_{m-1} + O(m^{3m+2})$ time. From these, we obtain ranking and unranking algorithms of $C(n, m)$, which runs in $t_m = t_{m-1} + O(m^{3m+2})$ time. Obviously we can let $t_1 = O(1)$. In consequence, we have $t_m = O(m^{3m+3})$. □

4 Concluding remarks

We have presented two unranking algorithms of $C(n, m)$. One algorithm runs in $O(m \log n)$ time, and the other runs in $O(m^{3m+3})$ time. In particular, the running time of the latter algorithm depends on m only. This running time is evaluated by the number of arithmetic operations on numbers represented by $O(m \log n)$ bits. In a standard word-RAM model, it may be usual to suppose that the word-size is $O(\log n)$. Even in this model, the running time of this algorithm depends on m only.

An obvious future work is to achieve better running time. For permutations of m-element subsets chosen from an n-element set, $O(m)$-time ranking and unranking algorithms are proposed by [18, 19]. Hence the existence of $O(m)$-time ranking and unranking algorithms for $C(n, m)$ is an interesting open problem.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo. Fast discovery of association rules. Advances in Knowledge Discovery and Data Mining, pp. 307–328, 1996.

[2] R. Agrawal, R. Srikant. Fast algorithms for mining association rules in large databases. Very Large Data Bases Conference '94, pp. 487–499, 1994.

[3] E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. SIAM Journal on Computing 2:1–6, 1973.

[4] R. Aringhieri, P. Hansen, F. Malucelli. Chemical trees enumeration algorithms. 4OR 1:67–83, 2003.

[5] C.J. Colbourn, R.P.J. Day, L.D. Nel. Unranking and ranking spanning trees of a graph. Journal of Algorithms 10:271–286, 1989.

[6] P. Eades. An algorithm for generating subsets of fixed size with a strong minimal change property. Information Processing Letters 19:131–133, 1984.

[7] T. Eiter, K. Makino. On computing all abductive explanations from a propositional Horn theory. Journal of the ACM 54:24, 2007.

[8] U.I. Gupta, D.T. Lee, C.K. Wong. Ranking and unranking of B-trees. Journal of Algorithms 4:51–60, 1983.

[9] T. Imada, S. Ota, H. Nagamochi, T. Akutsu. Enumerating stereoisomers of tree structured molecules using dynamic programming. The 20th International Symposium on Algorithms and Computation, LNCS 5878, pp. 14–23, 2009.

[10] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, K. Makino. Enumerating spanning and connected subsets in graphss and matroids. Journal of the Operations Research Society of Japan 50:325–338, 2007.

[11] D.E. Knuth. Generating all combinations and partitions, The Art of Computer Programming, Volume 4, Fascicle 3, 2005.

[12] Z. Kokosinskiński, Unranking combinations in parallel, 2nd International Conference "Parallel and Distributed Processing Techniques and Applications," pp.79–82, 1996.

[13] Z. Kokosinskiński. Algorithms for unranking combinations and other related choice functions. The University of Aizu, Technical Report 95-1-006, 1995.

[14] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan. Generating all maximal independent sets, NP-hardness and polynomial-time algorithms. SIAM Journal on Computing 9:558-565, 1980.

[15] J. Liebehenschel. Ranking and unranking of lexicographically ordered words: an average-case analysis. Journal of Automata, Languages and Combinatorics 2:227–268, 1998.

[16] H. Liu, J. Wang. A new way to enumerate cycles in graph. Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, pp. 57, 2006.

[17] K. Makino, T.Uno. New algorithm for enumerating all maximal cliques. 9th Scandinavian Workshop on Algorithm Theory, LNCS 3111, pp. 260–272, 2004.

[18] M. Mareš, M. Straka. Linear-time ranking of permutations. 15th Annual European Conference on Algorithms, LNCS 4698, pp. 187–193, 2007.

[19] W. Myrvold, F. Ruskey. Ranking and unranking permutations in linear time. Information Processing Letters 79:281–284, 2001.

[20] J.B. Remmel, S.G. Williamson. Ranking and unranking trees with a given number or a given set of leaves. arXiv:1009.2060, 2010.

[21] F. Ruskey, A. Williams. Generating combinations by prefix shifts. 11th Annual International Conference on Computing and Combinatorics, LNCS 3595, pp.570–576, 2005.

[22] B. Selman, H. J. Levesque. Support set selection for abductive and default reasoning. The International Journal of Artificial Organs 82:259–272, 1996.

[23] T. Takaoka, S. Violich. Fusing loopless algorithms for combinatorial generation. International Journal of Foundations of Computer Science 18:263-293,2007.

[24] T. Uno. A fast algorithm for enumerating bipartite perfect matchings. The 12th International Symposium on Algorithms and Computation, LNCS 2223, pp. 367–379, 2001.